1. Beadandó feladat (Négyzetek) dokumentáció

Készítette:

Vörös Döme

Neptun-Kód: QK8IUC

E-mail: qk8iuc@inf.elte.hu

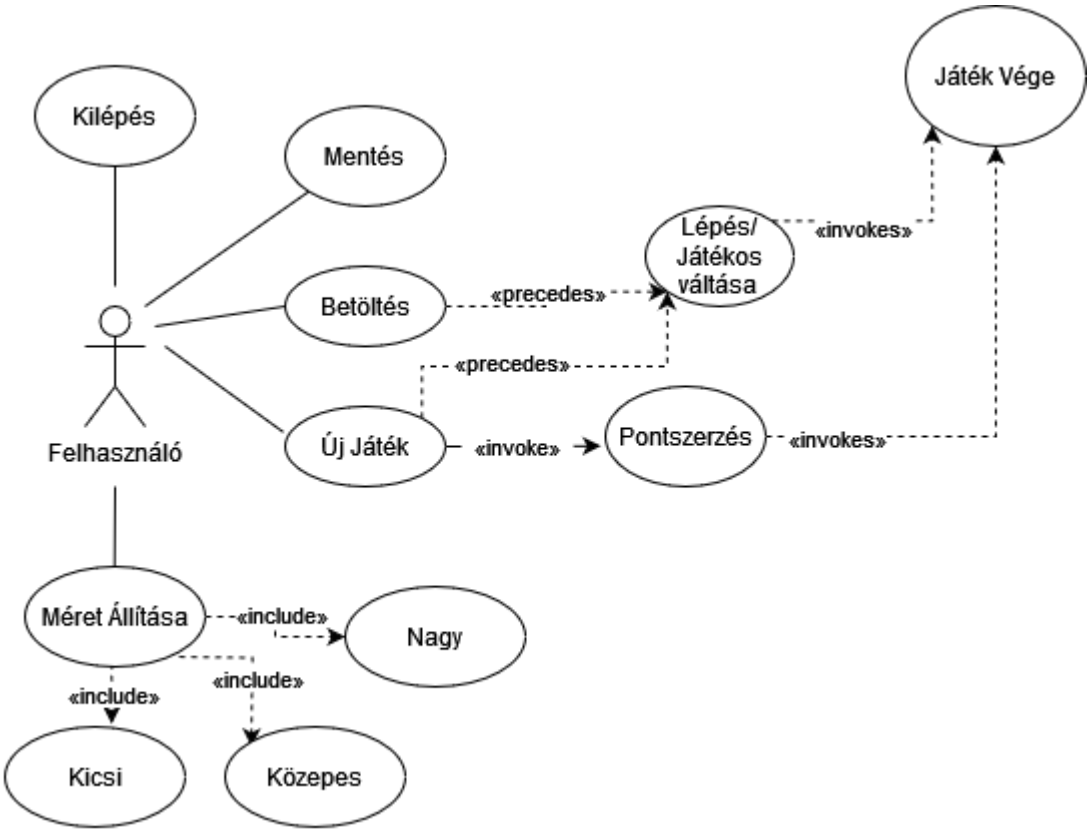
Feladat:

Készítsünk programot, amellyel az alábbi két személyes játékot játszhatjuk. Adott egy $n \times n$ pontból álló játéktábla, amelyen a játékosok két szomszédos pont között vonalakat húzhatnak (vízszintesen, vagy függőlegesen). A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta). Ilyen módon egyszerre akár két négyzet is elkészülhet. A játék addig tart, amíg lehet húzni vonalat a táblán. A játékosok felváltva húzhatnak egy-egy vonalat, de ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik. A program biztosítson lehetőséget új játék kezdésére a pályaméret megadásával (3×3, 5×5, 9×9), játék mentésére és betöltésére. Ismerje fel, ha vége a játéknak, és jelenítse meg, melyik játékos győzött (ha nem döntetlen). Játék közben a vonalakat, illetve a négyzeteket színezze a játékos színére.

Elemzés:

- A játékot három nehézségi szinttel játszhatjuk: könnyű (3x3as tábla), közepes (5x5ös tábla), nehéz (9x9es tábla). A program indításkor könnyű szintet állít be, és automatikusan egy új játékot indít el.
- A feladatot egyablakos asztali alkalmazásként Windows Forms grafikus felülettel valósítjuk meg.
- Az ablakban elhelyezünk egy menüt a következő menüpontokkal: File (New Game [Új Játék], Load Game [Játék Betöltése], Save Game [Játék Mentése], Exit Game [Kilépés])
- A játéktáblát 3 különböző rács reprezentálja. Az első amely olyan nagy mint a tábla a mezőket jelzi amiket beszínezünk ha a 4 vonal körülötte be lett színezve. És van egy horizontális és vertikális oszlop is amik a vonalakat fogják jelképezni.
- A játék automatikusan feldob egy dialógusablakot, amikor vége a játéknak (nyert valamelyik játékos vagy döntetlen lett) vagy amikor ki akarunk a programból lépni. Szintén dialógusablakokkal végezzük el a mentést, illetve betöltést, a fájlneveket a felhasználó adja meg.

- A felhasználói esetek az 1. ábrán láthatóak.



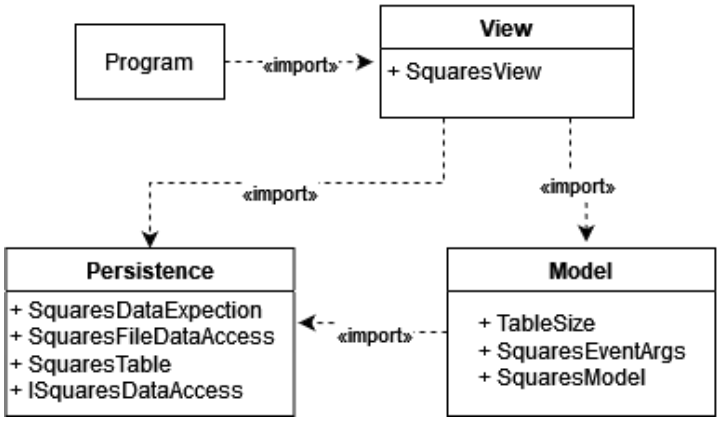
1. ábra: Felhasználói esetek diagramja

Tervezés:

- Programszerkezet:
  - A programot háromrétegű architektúrában valósíjuk meg. A megjelenítés a View, a modell a Model, míg a perzisztencia a Persistence névtérben helyezkedik el. A program csomagszerkezete a 2. ábrán látható.
  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a Persistence és Model csomagok a program felületfüggetlen projektjében, míg a View csomag a Windows Formstól függő projektjében kap helyet.
- Perzisztencia:
  - Az adatkezelés feladata a Squares táblával kapcsolatos információk tárolása, valamint a betöltés/mentés biztosítása.
  - A **SquaresTable** osztály egy érvényes Squares táblát biztosít (azaz mindig ellenőrzi a beállított értékek), ahol mind a 3 mezőcsoportnak ismert az Field (**Empty**, **Player1**, **Player2**) értéke (**_Squares**, **_Horizontal**, **_Vertical**). A játék kezdetekor mindegyik mező **Empty** értéket fog kapni. A tábla alapértelmezés

szerint 3 x 3 -as alapmező és 3 x 4 és 4 x 3-as horizontális és vertikális mezőkkel rendelkezik, de ez a konstruktorban paraméterezhető. A tábla lehetőséget az állapotok lekérdezésére (**IsFilled**, **GetTableValue**, **IsSpaceFilled**), direkt beállítás (**SetTableValue**) elvégzésére, illetve a tábla elemeinek újra beállítására, amelyet egy játék indításakor alkalmazunk (**ResetTable**).

- A hosszú távú adattárolás lehetőségeit az **ISquaresDataAccess** interfész adja meg, amely lehetőség ad a tábla betöltésére (LoadAsync), valamint mentésére (SaveAsync). A műveleteket hatékonysági okokból aszinkron módon valósítjuk meg
- Az interfészt szöveges fájl alapú adatkezelésre a **SquaresFileDataAccess** osztály valósítja meg. A fájlkezelés során fellépő hibákat a **SquaresDataExpection** kivétel jelzi.
- A program az adatokat szöveges fájlként tudja eltárolni, melyek az **sqr** kiterjesztést kapják. Ezeket az adatokat a programban bármikor be lehet tölteni, illetve ki lehet menteni az aktuális állást.
- A fájl első sora megadja a tábla méretét, az első és második játékos pontszámát, illetve az aktuális játékos számát (1 – első játékos, 2 – második játékos). A fájl többi része izomorf leképzése a játéktábla 3 különböző részének azaz kis tábla esetén 10 sor, a közepes tábla esetén 16 sor, a nagy tábla esetén 28 sor következik, melyben az aktuális mezőnek van leírva a Field állapota (üres, első játékosé, második játékosé). A számok 0-2 lehetnek, ahol a 0 az üres mezőt, az 1 a első játékos mezőjét, a 2 pedig a második játékos mezőjét adja meg.

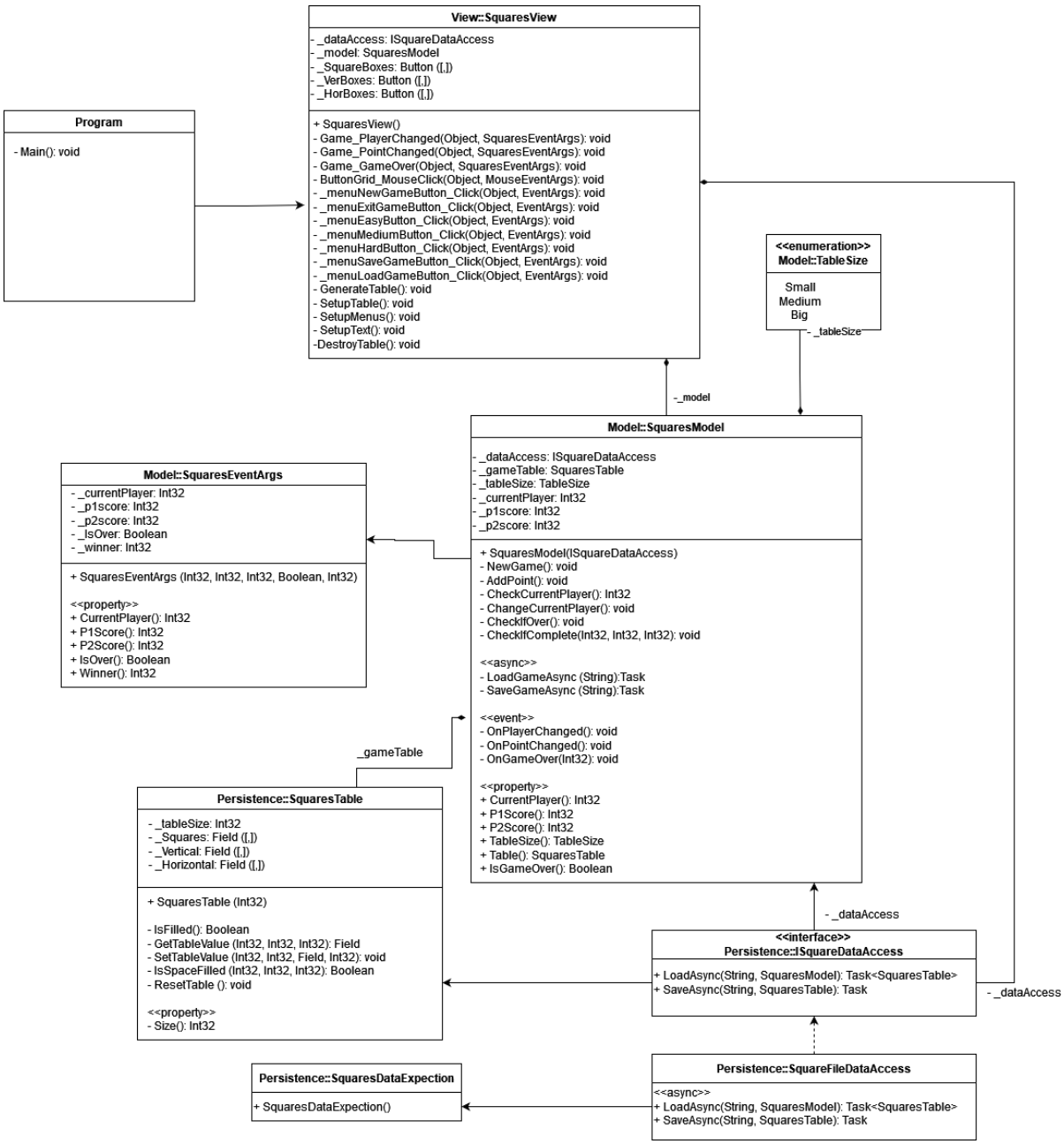


2. ábra: Az alkalmazás csomagdiagramja

- Modell:
  - A modell lényegi részét a **SquaresGameModel** osztály valósítja meg, amely szabályozza a tábla tevékenységeit, valamit a játék egyéb paramétereit, úgymint a két játékos pontszáma (**_p1Score**, **_p2Score**) és az aktuális játékost (**_currentPlayer**). A típus lehetőséget ad új játék kezdésére (**NewGame**), pontszám hozzáadáshoz (**AddPoint**), aktuális játékos lekérdezésére (**CheckCurrentPlayer**) és megváltoztatására

(**ChangeCurrentPlayer**), a játék végének ellenőrzésére (**CheckIfOver**) és egy adott négyzet ellenőrzésére, hogy be lehet-e színezni (**CheckIfComplete**)
    - A játékállapot változásáról (pontszám változás, játékos változás) a **PointChanged** és **PlayerChanged** esemény, míg a játék végéről a **GameOver** esemény tájékoztat. Az események argumentuma (**SquaresEventArgs**) tárolja a győzelem állapotát, a pontszámokat, és az aktuális játékost.
    - A modell példányosításkor megkapja az adatkezelés felületét, amelynek segítségével lehetőséget ad betöltésre (**LoadGameAsync**) és mentésre (**SaveGameAsync**)
    - A játéktábla méretét a **TableSize** felsorolási típuson át kezeljük, és a **SquaresGame** osztályban konstansok segítségével tároljuk az egyes nehézségek paramétereit.
- Nézet:
    - A nézetet a **SquaresView** osztály biztosítja, amely tárolja a modell egy példányát (**_model**), valamit az adatelérés konkrét példányát (**_dataAccess**)
    - A játéktáblát egy dinamikusan létrehozott gombmező (**_buttonGrid**) reprezentálja. A felületen létrehozzuk a megfelelő menüpontokat, illetve státuszsort, valamit dialógusablakokat, és a hozzájuk tartozó eseménykezelőket. A játéktábla generálását (**GenerateTable**), illetve az értékek beállítását (**SetupTable**) külön metódusok végzik.

- A program teljes statikus szerkezete a 3. ábrán látható:



3. ábra: Az alkalmazás csomagdiagramja

Tesztelés:

- A modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **SquaresGameModelTest** osztályban.
- Az alábbi tesztesetek kerültek megvalósításra:
  - **SquaresGameModelNewGameEasyTest**
    SquaresGameModelNewGameMediumTest

SquaresGameModelNewGameHardTest: Új játék indítása, a mezők beállítása, a két játékos pontszáma és az aktuális játékos ellenőrzése, illetve a mezők megszámolása a táblaméret függvényében.

- **SquaresGameModelPlayersTest**: Új táblán mindkét játékos lép egyet, mely közben ellenőrzöm, hogy nem ad nekik pontszámot és hogy megfelelően váltogatja az aktuális játékost.
- **SquaresGameModelPointTest**: Új táblán beállítok pont annyi mezőt, hogy a következőnél teljes legyen a négyzet és a lépés előtt és után is tesztelem hogy a pontszámok megfelelően növekednek és hogy a teljes négyzet miatt az aktuális játékos mégegyszer léphet majd.
- **SquaresGameModelTwoPointTest**: Új táblán beállítok pont annyi mezőt, hogy a következőnél kettő négyzet is teljes legyen és a lépés előtt és után is tesztelem hogy a pontszám 2-vel növekszik és hogy a teljes négyzet miatt az aktuális játékos mégegyszer léphet majd.